Journal of AI and Data Mining
Vol 8, No 2, 2020, 201-212.
DOI: 10.22044/JADM.2019.8768.2011

Community Detection using a New Node Scoring and Synchronous Label Updating of Boundary Nodes in Social Networks

M. Zarezadeh, E. Nourani* and A. Bouyer

Department of Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran.

Received 05 August 2019; Revised 09 October 2019; Accepted 21 December 2019
**Corresponding author: ac.nourani@azaruniv.ac.ir (E. Nourani.).*

Abstract
Community structure is vital to discover the important structures and potential properties of complex networks. In the recent years, the increasing quality of local community detection approaches has become a hot spot in the study of complex networks due to the advantages of linear time complexity and applicable for large-scale networks. However, there are many shortcomings in these methods such as instability, low accuracy, and randomness. The G-CN algorithm is one of the local methods that uses the same label propagation as the LPA method but unlike LPA, only the labels of boundary nodes are updated at each iteration that reduces its execution time. However, it has a resolution limit and a low accuracy problem. In order to overcome these problems, this paper proposes an improved community detection method called SD-GCN, which uses a hybrid node scoring and synchronous label updating of boundary nodes along with disabling random label updating in initial updates. In the first phase, it updates the label of boundary nodes in a synchronous manner using the obtained score based on the degree centrality and common neighbor measures. In addition, we define a new method for merging communities in a second phase, which is faster than the modularity-based methods. Extensive set of experiments are conducted to evaluate the performance of SD-GCN on small- and large-scale real-world networks and artificial networks. These experiments verify a significant improvement in the accuracy and stability of community detection approaches in parallel with a shorter execution time in a linear time complexity.

Keywords: *Social Networks, Community Detection, Boundary Node, Label Propagation.*

1. Introduction

A wide range of complex systems can be represented by networks such as many biological networks, i.e. protein-protein interaction network, power grids, transportation systems, and social network. These networks consist of nodes and links [1, 2]. Typically, these networks are represented by a complex relational graph. In social networks, a node is an entity such as people, organizations, and companies. A link is defined as a type of relationship among nodes. By analyzing the social networks, an important structural property can be seen that is called the community structure [3]. The communities are groups of nodes where the connections inside a group are dense but between which they are sparser [4]. Many important structural features of a social network can be extracted based on the structure of communities. Community detection is a process to recognize such groups of nodes in a network. Many studies have been conducted to extract communities within a network; however, most of them are not applicable to large networks due to high orders of time complexity.

As the size of complex networks grows, the need for algorithms with a low time complexity becomes more important. The community detection algorithms are generally divided into three categories: global, semi-local, and local. Global methods require the access to the whole network that need a high computational time and is fundamentally impossible for a large network [5]. Therefore, the local and semi-local community detection algorithms offer a practical alternative [6]. Various local methods have been proposed that use the topological properties of the network to detect the community structures in a local manner.

Label Propagation Algorithm (LPA) is one of the basic and fast community detection methods for large-scale networks on account of its near-linear time complexity [7]. This method starts with assigning an initial label to each node. This label will be replaced by the most frequent label of neighbors within every iteration. A random label is selected when frequencies are equal. The iterations are continued until a consistent state is achieved. Finally, the nodes that share the same label are considered as a community. LPA is simple and acts as a base for many other approaches.

Another approach for large networks is called the G-CN method [8]. Similar to LPA, this method is also based on label propagation but, in contrast to LPA, it updates only boundary nodes in every iteration based on the number of common neighbors, not the frequency of labels. This leads to a fast running time and a performance better than LPA [8]. However, this method suffers from the instability in producing the final due to randomly updating nodes' label. Another problem with G-CN is its weak performance in detecting an appropriate number of communities. G-CN detects the small and local communities [9].

In order to overcome these problems, we present SD-GCN, which applies a two-phased approach for community detection. The first phase tries to improve the results of G-CN using a new updating strategy and without increasing the linear order of complexity. It requires a fewer number of the label updates, which leads to the stability of results and shortening the execution time. In the second phase, a new measure is used to integrate the communities in a suitable manner. In the proposed algorithm, we present a synchronized and informed updating method during the initial updates.

After providing a brief overview, we introduce some related works in Section 2. In Section 3, we present our proposed algorithm in more detail. Thereafter, the experimental results on the synthetic and real world networks are shown. Lastly, the final section offers the concluding remarks and further directions.

2. Related works

In the recent years, community detection in complex networks, principally in social networks has been the subject for wide research studies. There are many different groups of methods for community detection such as clustering-based methods [10,11], modularity-based methods [12, 13], spectral clustering algorithm [14], dynamic algorithm [15, 16], label propagation [7,17,18], and similarity-based methods [17,19,20]. For example, modularity-based methods try to optimize the modularity measure during the identifying and expanding communities. For instance, Newman has proposed a greedy optimization method for a modularity maximization problem [13]. In this method, each node is supposed to be a different community with one member. Then the pair of communities with the largest increase (or smallest decrease) in modularity merges repeatedly until all the nodes are merged together in a sole community. In label propagation-based algorithms, each node has an initial label. Afterwards, in each iteration, the label of each node is updated based on the most frequent label of its neighbors.

On the other hand, the community detection algorithms can be divided into the overlapping and non-overlapping algorithms. In the non-overlapping community detection approaches, a node is only assigned to an individual community [21]. In addition, the community detection algorithms can be used in local and global manners. In local methods, only the information about a node and its neighbors is used to discover a community. However, the global methods require a comprehensive information about all nodes or links for assigning a node to a community [4,22]. This method has a better performance but requires a high time complexity that makes them infeasible in large-scale networks. Therefore, in this paper, we only focus on the local or semi-local methods due to their fast detection. Many different local algorithms have been proposed in the recent years. As instance, the LPA−CNP algorithm [23] developed a new strategy for label updating of LPA by considering the effect of neighbors that are more than one hop away. In the COPRA algorithm, a global parameter is used to allow each node to carry multiple labels [24]. Thus each node could belong to multiple communities, and therefore, COPRA can find an overlapping community structure. The CK−LPA algorithm [25] is a new development of LPA based on the core expanding procedure that weights each link based on a local similarity before expanding. During expanding and label updating, this weight is used in tie break states. The LPA−S algorithm uses a synchronous label propagation strategy by the probability of each surrounding label to update labels. The network weighting strategy has been claimed to be useful in [26] by adding a pre-processing step and edge weights according to their centrality by performing multiple random walks on the network. In this method, the centrality of an edge reflects its contribution to spreading messages over the network. Recently, an integer linear programming-based model [27] has been proposed to detect the community structure and also identify the most influential nodes within

each community. Another recent approach [28] has proposed a new variant of the variable Neighborhood Decomposition Search (VNDS) heuristic for solving global optimization problems and applies it in detecting communities in large networks using modularity maximization.

The G-CN algorithm uses the label propagation method and only updates labels of boundary nodes [8]. For label selection, a maximum number of common neighbors is considered. In fact, G-CN uses a local similarity measure based on the common neighbors to calculate a score for identifying the community of boundary nodes [8]. DCNR is an extended version of G-CN that combines common neighbors and node degree local measures to improve the performance of G-CN [9]. However, DCNR has some problems in updating labels and merging process. In this paper we propose the SD-GCN algorithm that is based on DCNR and presented in two phases. Because of the considerable improvement in the first phase, the next phase in this algorithm is of low importance. In the proposed method, label updating is in the synchronize strategy and avoids random updating in the initial steps of the algorithm.

3. Proposed algorithm

The proposed algorithm is a new development of the DCNR and G-CN algorithms that is called SD-GCN. SD-GCN has been designed to overcome the shortcoming of the G-CN method. The label propagation-based algorithms are mostly iterative and update all labels in each iteration. However, G-CN only updates the labels of boundary nodes during an iteration in an asynchronous manner. Boundary nodes are located in the border of communities, and therefore, are recognizable from the internal nodes. The internal node is a node that has the same label with all its neighbors, and the boundary node is a node that is not an internal node. G-CN categorizes the neighbors of a node based on their labels, and assigns a utility score to each community in the neighborhood. The label of the node will be the same as the community with the maximum utility score. Utility score is the number of common nodes with the neighboring communities. In contrast, LPA assigns the label based on the most frequent label in the neighborhood nodes.

G-CN has a linear order of execution time like LPA, and therefore, is suitable for large networks. Besides having better results, G-CN is faster than LPA thanks to updating only the boundary nodes. In SD-GCN, both the common neighbor measure and degree centrality are used to compute the utility score. Along with other modifications to standard G-CN, our approach leads to significantly improved and stable results with a shorter execution time. SD-GCN consists of two main phases, as follow.

3.1. First phase of SD-GCN

This phase is similar to G-CN with a difference in the computing utility score. Here, the score is based on the number of common neighbors with communities and the degree of neighbor nodes. Considering the degree centrality as a local feature, SD-GCN achieves a clear improvement in the results. Assume G(V, E) is an undirected, unweighted, and acyclic graph, where V and E are the node and edge sets accordingly. V = {v_1, v_2, ..., v_n} and E = {(v_i, v_j)| v_i, v_j ∈ V}, where m = |E| is the number of edges and n = |V| is the number of nodes. Therefore, the common neighbors of the nodes I and j can be computed as follows:

$$B_i(j) = \left| N_i \cap N_j \right| \quad (1)$$

where, N_i and N_j are the neighbors of the nodes I and j, respectively.

The degree centrality of a node is the number of relations of a node with other nodes.

$$Deg(V_i) = \text{number of neighbors of } V_i \quad (2)$$

In order to update the label of a node, its common neighbors are categorized based on their labels. Then a utility score is computed for each neighbor community using Equation (3). This score is similar to G-CN and equal to the sum of common neighbors.

$$S(k) = \sum_{j \in k} B_i(j) \quad (3)$$

In computing the utility score, sometimes we encounter similar scores, and we need to select randomly one of the communities. For instance, in the Zachary karate club (Figure 1), based on Equation 3, the label of the node 10 is selected randomly due to the mentioned problem. A wrong label selection, consequently causes assigning wrong labels to the neighbors. In the worst random case, the nodes 25 and 26 will be considered as a separate community since they have only node 32 as the common neighbor.

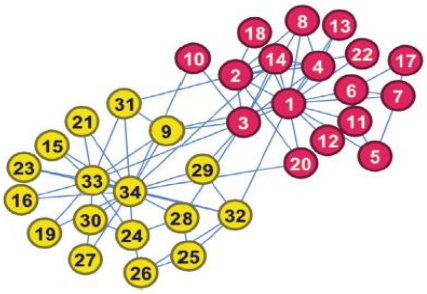

Figure 1. Zachary Club Karate network.

In order to overcome this challenge, we compute the utility score for community K in the neighboring of node j by Equation (4):

$$S(k)=(\frac{E}{N}\times\sum_{j\in k}B_i(j))+\sum_{j\in k}Deg(V_j) \quad (4)$$

where, the score is based on the total number of common neighbors along with the other local feature, which is the sum of the node degrees within community K. Also N is the total number of graph nodes. Clearly, the value resulting from aggregating the number of common neighbors is relatively small in comparison with the sum of the node degrees in the second part of the equation. In order to balance the effects of both part of the equation, we multiply E/N by $S(k)$, where E/N is the average degree for graph nodes.

The label of a node is determined based on the label of a community that has the maximum utility score. The new approach for utility computation has improved results, while requiring a fewer label updates that leads to a more stable result. On the other hand, decreasing the number of random updates causes a faster formation of communities with a shorter execution time. Our approach utilizes only local features including the number of common neighbors and the degree node centrality. Therefore, the time complexity of the first phase is O(n). Furthermore, thanks to updating, only the boundary nodes, our method is faster than LPA.

SD-GCN starts with the label initialization of all nodes done. Then the boundary nodes are determined and placed in set B. In an iterative process, the label of each boundary node in set B will be updated, and consequently, its non-boundary neighbors will be checked. If any neighbor becomes a boundary node, it will be placed in B. The algorithm stops when there is no boundary node in set B. Our experiments reveal that the asynchronous label updating in G-CN can be replaced by an asynchronous method to improve the results. In the asynchronous label updating, the labels of the previous iteration are considered, whereas in the synchronous behavior, the updated label has an effect on the computations of the current iteration. The consequent nodes in set B are the likely neighbors of each other because during the label change of a node, its neighbors are added to B if they become a boundary node. Therefore, if the synchronous method is applied, every change in a node immediately has an effect on the consequent neighbor computations. Hence, the synchronous approach is more suitable for updating the boundary nodes. Our experiments verify this idea by exhibiting considerable improvements along with decreasing the random update in comparison with the asynchronous manner, although our approach tries to avoid a random update during the initial update steps. By disabling random updates, we achieve a considerable improvement and stable results. This also leads to a shorter execution time because of a faster community formation.

To sum up the first phase, we improved G-CN by utilizing the degree centrality, changing the approach to synchronous and avoiding random update during the initial update. A flowchart of the first phase is presented in figure 2.

The algorithm starts with an initial update, which initializes all nodes with individual labels. Then all labels of nodes are updated. After that, the boundary nodes are determined and placed in set B. If there is no label updating, all nodes are added to set B. As discussed above, the algorithm continues by selecting one node from B and updating its label. If there is a change in the label of node, all its neighbors will be checked again, and if they become a boundary node, they are inserted in set B. The iterations are terminated when there is no more node in set B.

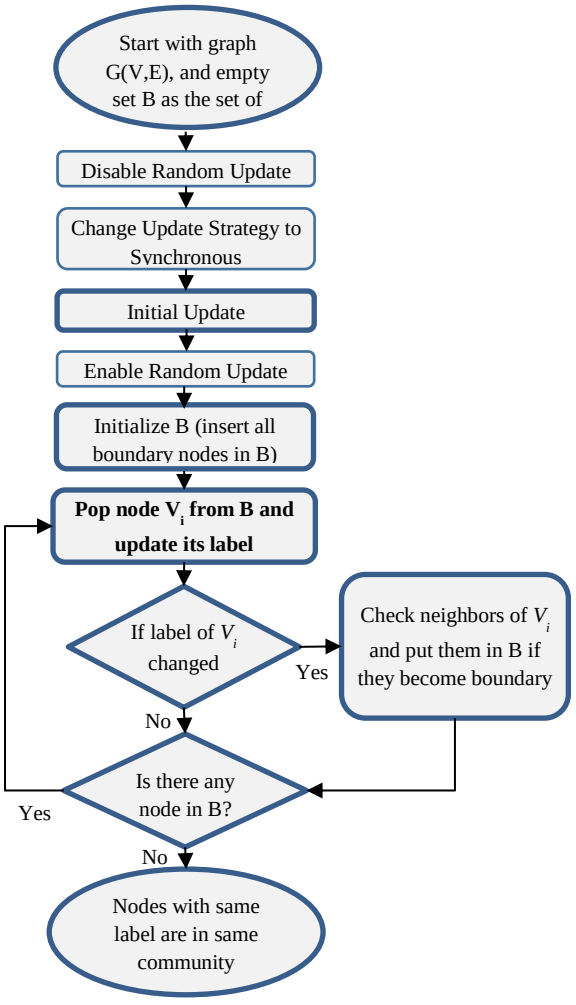

Figure 2. Flowchart of phase 1 in the proposed SD-GCN method.

Algorithm 1 is the first phase of the proposed method. Function initial_update updates the labels of all nodes in one iteration, and thus some nodes may become an internal node, and this function avoids considering all nodes as a boundary node, and the number of nodes in B (boundary nodes set) will be decreased.

3.2. Second phase: Merging communities

Although the result of the first phase has considerable improvements in comparison with G-CN, it is also extended to overcome the problem of solving small communities. There are many small communities in the G-CN algorithm, and the number of generated communities are more than the actual ones in the ground truth. Therefore, in this phase, some of them will be merged to tackle this challenge. We propose a new density-based method for merging communities in contrast to the other approaches that try to utilize modularity optimization for merging community because modularity optimization has a more waste considerable execution time than our merging method.

Density is computed as follows:

$$\text{Density}(k) = \frac{\text{number of edges}}{n(n-1)/2} \tag{5}$$

The number of required edges for integration is presented in Equation 6.

$$f = \frac{\sqrt{n_x}}{\sqrt{n_y}} \times \frac{D_p}{D_q} \times m_z \frac{(D_1 + D_2)}{2} \tag{6}$$

where, $\frac{(D_1 + D_2)}{2}$ is the average density of two communities, m_z is the minimum edge number, D_p and D_q are the maximum and minimum density values, respectively, and n_x and n_y are the minimum and maximum numbers of degrees of the two communities. If we assume that n1 and n2 are the numbers of nodes in the c1 and c2 communities, D1 and D2 are their densities, and m1 and m2 are the numbers of the edges of the c1 and c2 communities, respectively.

$$\{m_z = \min(m_1, m_2) \tag{7}$$

$$\begin{cases} D_p = \max(D_1, D_2) \\ D_q = min(D_1, D_2) \end{cases} \tag{8}$$

$$\begin{cases} n_x = min(n_1, n_2) \\ n_y = max(n_1, n_2) \end{cases} \tag{9}$$

The expected number of edges between the two communities is denoted by *f* and the current number of edges is presented by *z*. Therefore, we have:

z = number of edges between c_1 and c_2 (10)

$$d(c_1, c_2) = f - z \tag{11}$$

Based on Equation 6, the difference between the numbers of nodes in the two communities leads to a decrease in the required number of edges for merging, whereas the difference in density between the two communities leads to increasing the *f value*, which prevents integrating a dense community with a sparse one. The modularity optimization-based methods struggle with the resolution limit challenge because modularity of two communities with extremely different sizes may lead to integrating them. The second phase of the proposed approach solves this problem. Furthermore, the monster community is another challenge of local algorithms like LPA and its variants, in which really individual communities combine with each other and form a monster community. This problem particularly happens for small communities. The SD-GCN method does not have these challenges.

However, it has a problem for some small and sparse communities that is negligible in most of the time.

The steps of the second phase are as follow:

1- Difference between the current number of edges and the required number edged for integration (based on Equation 11, this difference is called *d*) is computed for any pair of communities and the lowest difference is determined.
2- Two communities with the minimum *d* are merged; refresh the list with the updated *d* values taking into account the new generated community.
3- If the list of *d* values is empty, the algorithm is finished; otherwise, go to step 2.

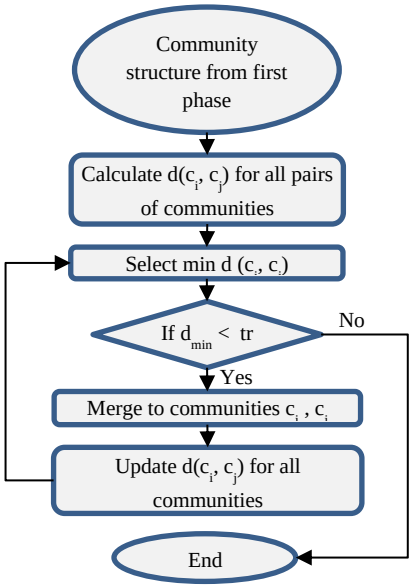

Figure 3. Flowchart of phase 2 in the SD-GCN method.

Algorithm 1. **First phase of the proposed method**

```
# V: set of nodes
# L[i]: label of Vi
# B: set of boundary nodes
# Best_Community (Vi): Finds the best community for node Vi according to Equation (4).
# Initial_Update: Calls the Best_Community function for all nodes for label initialization
Input : Adjacency list of G(V,E) as undirect and unweighted graph
Output : L (labels of nodes)
{
    Disable random update
    Initial_update()
    Initialize_B() #Initialize the set of boundary nodes for first
    Enable random update
    While B != ∅
        vi = randomly selected node from B
        remove I from B
        L[i] = best community(vi): update label of vi
        If label of vi is changed then
        Check neighbors of vi and push them to B if became a boundary node
    Return L
}
```

Algorithm 2 **Second phase of the proposed method**

```
# C: Extracted community structure in first phase
# d(ci, cj): Difference between the current number of edges and required number edged for integrating Ci and Cj, Equation (11)
# Merge(Ci, Cj): merge community Ci and Cj
Input: Extracted community structure in first phase
Output: Final community structure
{
    Calculate d(Ci, Cj) for all pairs of communities
    While true
        Select minimum value of d(Ci, Cj)
        If d < threshold
            Break
        Merge(Ci, Cj) # two communities with minimum value of d is merged
        Calculate d between new merged community and other ones
    Return C # final community structure (some communities are merged in C )
}
```

A flowchart of the second phase is presented in figure 3. In this flowchart, *tr* is the threshold value and is equal to 0 in our experiments.
Algorithm 2 is the second phase of SD-GCN.

3.3. Complexity analysis

3.3.1. First phase

In this phase, first, the labels of all nodes are updated in the Initial_Update() function whose time complexity is O(n). Then all the boundary nodes are detected and pushed to set B. The number of boundary nodes is less than N (total number of nodes), and in the community detection process, only a few times is possible that a node is added to the boundary node set. Thus the time complexity of the first phase is O(n).
In terms of memory usage, all the data structures that are used are the adjacency list of nodes, list of labels, and boundary node set. Thus the memory usage has a linear relation with the total number of nodes (N).

3.3.2. Second phase

The time complexity of the second phase is computed as follows. We assume k as the number of all the detected communities by phase 1, n as the number of all nodes, and E as the number of all edges. Difference d is computed in a nested loop. which iterates k*k time and each iteration, density Therefore, the time complexity calculating the required number of edges for integration will be as follows:

$$T = k^2\left(\frac{2n}{k}+\frac{2E}{k}+\frac{k^2 n}{k^2}\right) = 2k\left(n+E\right)+k^2 n \qquad (12)$$

After any integration, only the updated values for the new community and others will be refreshed, and recalculating all pairs is not required. The order of time complexity in this phase is O(*kE*). Clearly, the complexity of this phase is higher than the first phase, and it is recommended to be used only for small communities.

Memory usage for the second phase is very little and the community structure from the first phase is considered to merge some communities if required.

4. Experiments

We evaluated SD-GCN over several real-world and artificial datasets. The real-world datasets include four small- and three large-scale ground truth datasets.

4.1. Datasets

In this section, we summarize the datasets used in our experiments for evaluating the proposed method.

4.1.1. Small real-world datasets with ground truth

We start the evaluations with real-world problems. Characteristics of the three utilized small dataset are summarized.

- *Zachary karate club network* is a very popular network referenced in several methods [26]. This network includes 34 nodes and 78 edges that represent associations between members of the club at a university in the United States. In the Zachary's karate club, all nodes are included in two communities.
- *Dolphin network* is derived from Lusseau's study about the behavior of 62 bottlenose dolphins living in New Zealand's Doubtful Sound [27]. The dolphin network includes 62 nodes and 159 edges. The nodes represent the 62 dolphins and each edge represents the interaction between the two dolphins.
- *US College Football network* shows the games' schedule among Division I of the US College Football League during the 2000 season [3, 28]. This network consists of 115 teams (nodes) and 613 edges. Each edge connects two teams that play together in this league. The college football network is divided into 12 communities. This database is notated as Football in the following tables.
- *Political Books network* is composed of American politics that compiled by Valdis, Krebs during the 2004 presidential election. This network involves 105 books (nodes) and 441 edges. Each edge connects two nodes whose related books are purchased together. In this network, all nodes are divided into three communities [17].

4.1.2. Large-scale real-world datasets with ground truth

SD-GCN is also evaluated on large real-world networks with the ground-truth of their communities, provided by SNAP [29]. These datasets include Youtube, Amazon, and DBLP. Table 1 shows the details about these networks.

Table 1. Details of large real-world networks.

Network	# of Nodes	# of Edges	# of Communities
Youtube	1134890	2987624	8385
Amazon	334863	925872	75149
DBLP	317080	1049866	13477

4.1.3. real-world networks without ground truth

For further evaluation, the proposed method is tested on some real-world networks. The community structure of these networks are not available for the algorithm. Table 2 shows the details of these networks.

Table 2. Details of real-world networks without ground truth of communities.

Network	# of Nodes	# of Edges
Email	143	623
Power grid	4941	6594
Face book	4039	88234
Twitter	404719	713319

4.1.4. LFR benchmark network

This network is computer-generated. We evaluate our method on six different LFR networks with a different number of nodes and parameters. Table 3 shows the details of the LFR networks. The average degree is denoted by k, *maxk* is the maximum degree, *minc* is the minimum number of communities, and *maxc* is the maximum number of communities. The mixing parameter μ is changed between 0.1 and 0.9 to produce 54 networks. For larger μ values, the community structures become less explicit.

Table 3. Details of LFR artificial networks.

Name	# of nodes	K	maxk	minc	maxc
LFR1	1000	15	20	20	45
LFR2	1000	10	30	15	40
LFR3	2000	15	20	10	30
LFR4	2000	10	25	15	40
LFR5	5000	10	10	10	30
LFR6	5000	15	20	20	40

4.2. Evaluation metrics

4.2.1. Normalized mutual information

One of the important metrics for evaluating the community detection algorithms is the NMI measure. The NMI value in the best case is equal to 1, where the detected community is exactly the actual structure in the ground truth. A zero value for NMI is the situation where the algorithm detects the whole network as a community.

We evaluate SD-GCN based on NMI. To commute the NMI value, a confusion matrix (N) is calculated. The rows in this matrix are the actual communities and the columns are the detected communities. Each entry N_{ij} is the number of nodes in the actual community i that is present in the detected community j. Equation 13 presents the computation of NMI.

$$NMI(A,B)=\frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B}N_{ij}\log\left(\frac{N_{ij}n}{N_{i.}N_{.j}}\right)}{\sum_{i=1}^{C_A}N_{i.}\log(\frac{N_{i.}}{n})+\sum_{j=1}^{C_B}N_{.j}\log\left(\frac{N_{.j}}{n}\right)} \quad (13)$$

where A is the actual community structure, B is the detected structure, c_a and c_b are the numbers of the actual and detected communities, respectively, n is the number of nodes in the network, N_i is the sum of the i_{th} row, and N_j is the sum of the j_{th} column in the matrix N.

4.2.2. Modularity

Modularity is a measure that evaluates the quality of a community detection approach on a network. Calculation of the modularity is carried out by Equation 14.

$$Q=\frac{1}{2m}\sum_{ij}\left(A_{ij}-\frac{d_i d_j}{2m}\right)\times\delta\left(c_i,c_j\right) \quad (14)$$

where, *m* represents the number of edges, A_{ij} is an adjacency matrix, where A_{ij} = 1 if node *i* is linked to node ;otherwise, A_{ij} = 0. $\delta(c_i, c_j)$ is a piecewise function defined as: if ci = cj , then δ(ci, cj) = 1; else, δ(ci, cj) =0.

The maximum value for the modularity is 1. The high-quality clustering approaches lead to discover better communities and achieve larger modularity values. Modularity compares the actual number of intracommunity edges with the expected number of edges in a random graph with the same degree distribution.

4.3. Evaluation results

The proposed algorithm is implemented in Python 3. All experiments are evaluated on a laptop with Intel core i5 CPU (2 core 2.67 GHz) and 4 gigabytes of RAM.

Tables 4 and 5 present the experimental results on small and large networks based on the NMI values and for our proposed SD-GCN method and other compared algorithms such as G-CN , DCNR, LPA , LVN [30], and Infomap [31].

Based on the results in table 4, the first phase of SD-GCN, which runs in O(n), surprisingly achieves comparable results with other algorithms that run in a higher time complexity. In all small networks, our approach outperforms other compared algorithms. In large-scale data sets, SD-GCN has the best performance in the Amazon and DBLP datasets. However, in the Youtube dataset, DCNR has the best and SD-GCN has the second best performance based on the NMI measure.

Based on the results in table 4, it seems that SD-GCN has lower modularity values than the other approaches; for example, for the Karate and Dolphins networks, some methods have higher values of modularity. However, it should be noted that the best-known modularity based on the ground truth of these communities is achieved by SD-GCN.

Table 4. Experiment results on small real-world networks with ground-truth communities.

Datasets	SD-GCN		DCNR		G-CN		LPA		LVN		Infomap	
	NMI	Q	NMI	Q	NMI	Q	NMI	Q	NMI	Q	NMI	Q
Karateh	**1**	0.371	1	0.371	0.847	0.378	0.753	0.379	0.59	**0.42**	0.7	0.401
Dolphins	**1**	0.379	0.7	0.392	0.552	0.473	0.74	0.412	0.48	**0.53**	0.5	0.527
Football	**0.927**	**0.601**	0.89	0.568	0.893	0.524	0.89	0.468	0.88	0.6	0.92	0.6
Polbooks	**0.576**	0.475	0.575	0.475	0.551	0.489	0.555	**0.554**	0.51	0.52	0.49	0.522

Table 5. Experimental results on large real-world networks with ground truth communities.

Datasets	SD-GCN		DCNR		G-CN		LPA		LVN		Infomap	
	NMI	Q	NMI	Q	NMI	Q	NMI	Q	NMI	Q	NMI	Q
Amazon	**0.694**	**0.793**	0.468	0.78	0.57	0.747	0.54	0.783	0.11	0.643	0.601	0.232
DBLP	0.545	**0.745**	0.349	0.734	0.56	0.713	0.64	0.674	0.13	0.696	**0.647**	0.714
Youtube	**0.109**	-	0.149	-	0.07	-	0.07	-	0.06	-	0.128	-

Table 6. Experimental results on real-world networks without ground-truth communities.

Datasets	SD-GCN	DCNR	G-CN	LPA	Infomap
	Q	Q	Q	Q	Q
Email	**0.473**	0.47	0.468	0.426	0.461
Power grid	**0.581**	0.52	0.492	0.581	0.572
Facebook	0.643	0.64	0.64	0.691	**0.703**
Twitter	**0.5**	0.488	0.50	0.50	0.496

Table 7. Execution time (ms).

Datasets	SD-GCN	DCNR	G-CN	LPA	Infomap
	T (ms)	T (ms)	T (ms)	T (ms)	T(ms)
Amazon	**107388**	108197	210237	578822	809408
DBLP	**168662**	169128	181155	494275	669057
Youtube	**1614414**	1622313	2310019	6075115	3518799

Table 8. Number of boundary nodes.

Network	SD-GCN	DCNR	GCN
Amazon	**321814**	323056	669726
DBLP	**297830**	298081	309849
Youtube	**1078145**	1080742	1440609

Therefore, the larger modularity values in the table are not the evidence of the superiority of the other approaches.

The results in table 5 show that SD-GCN achieves an improvement according to both the NMI and the modularity measures. Table 6 shows the results of the community detection algorithms on some networks that do not have ground truth for communities. Therefore, the NMI measure cannot be calculated for these networks, and the methods are evaluated only based on the modularity metric. The results show that our method outperforms the DCNR and GCN methods for all of datasets.

Also in the email and Power-Grid networks, the modularity values of SD-GCN are higher than the others.

In the Twitter network, the modularity values are approximately the same for all methods.

Table 7 presents the execution time of the experiments.

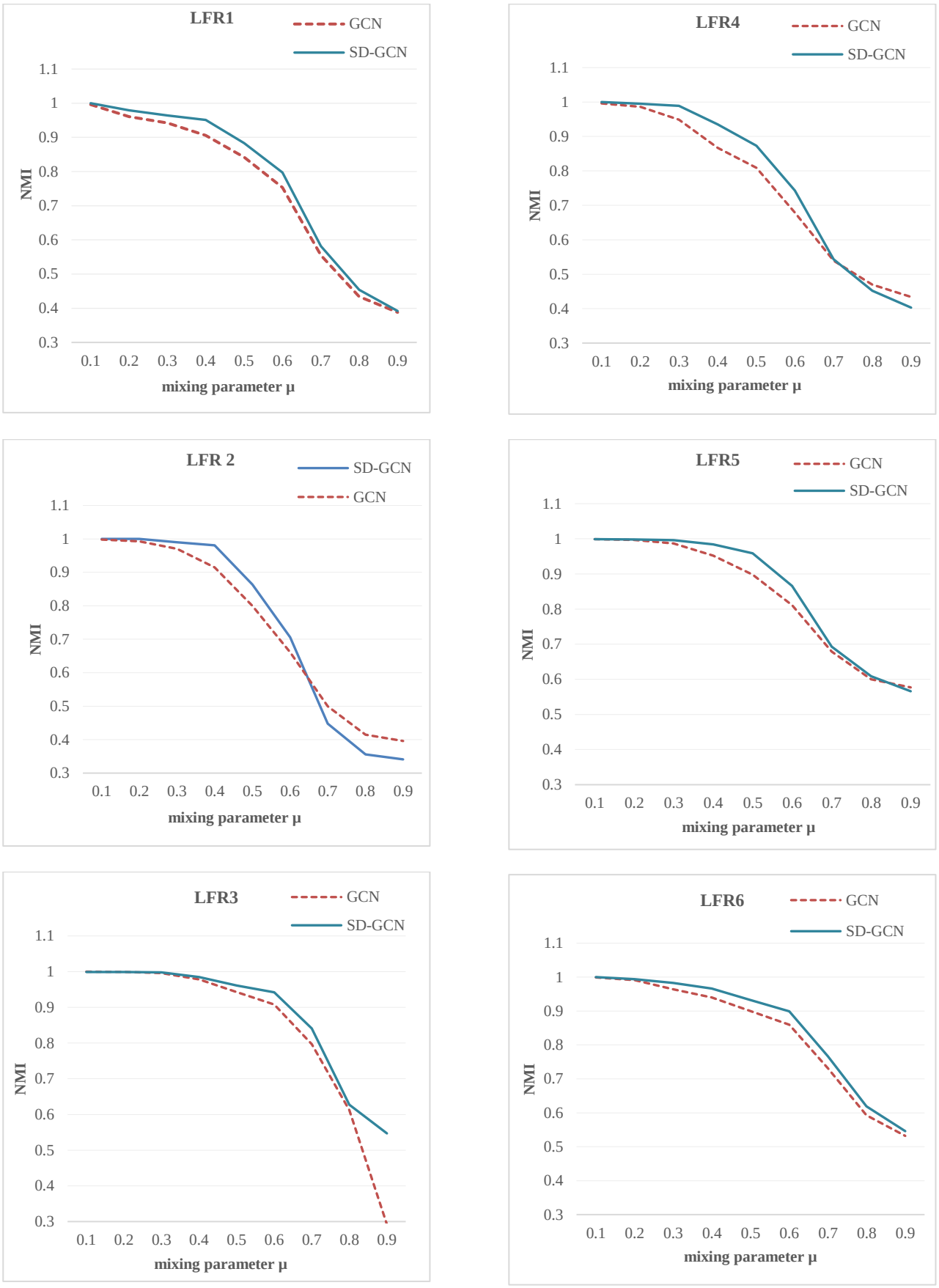

Figure 4. Experimental results on six LFR artificial networks for the SD-GCN and G-CN algorithms.

The test of execution time is only performed on large-scale datasets because the SD-GCN, G-CN, and DCNR, terminated in less than 1 ms, are considered as small datasets. These algorithms are also executed in the same condition. The execution time is depicted by T. According to table 7, the running time of SD-GCN is lower than all the other algorithms. G-CN is a recently introduced fast algorithm, and DCNR has a lower running time compared to G-CN. SD-GCN is faster than these algorithms because of a faster convergence of the label updating method based on a novel defined score.

Clearly, the lowest running time of the proposed approach is related to the number of label updates in the boundary nodes. It is clear that SD-GCN has the best execution time because of requiring less updates for the boundary nodes.

Table 8 shows the number of updated boundary nodes. Thanks to the lower number of label updates, we gain a lower running time.

For further experiments, we evaluate the proposed method over six LFR networks. The set of test is performed on LFR networks with 1000, 2000, and 5000 nodes. Figure 3 presents the experimental results over the LFR networks. Analysis of the results obtained shows that the proposed SD-GCN method has the best performance in $\mu = [0.1, 0.7]$. When $\mu = 0.1$, it means that the LFR network has a clear community. In this case, our algorithm does not have any wrong detection in all networks. When μ is greater than 0.5, the structures of the communities become increasingly unclear. However, the proposed SD-GCN method has a better detection than G-CN for $\mu > 0.5$ in all the LFR networks, except LFR2. In LFR2, G-CN has a better performance for $\mu > 0.6$. However, for $\mu > 0.7$ in the LFR network, it is not important due to their tendency to the random graphs.

5. Conclusion and future work

In this research work, we proposed a local community detection algorithm based on the G-CN algorithm called the SD-GCN algorithm. It includes two phases. Firstly, SD-GCN scores the nodes based on the degree centrality and common neighbor measures. Then a synchronized and informed updating method during the initial updates is used instead of the random labeling strategy. Therefore, SD-GCN requires a fewer number of the label updates, which leads to the stability of the results and shortening the execution time. In the second phase, a new measure is used to integrate the communities in a suitable manner. It can discover communities with a high quality regardless of the network size. On the small- and large-scale networks with both the clear and subtle community structures, SD-GCN outperforms other compared algorithms. This is due to the scoring and informed label updating method of this algorithm.

References

[1] Newman, M. (2018). Networks. Oxford university press.

[2] Dorogovtsev, S. N. & Mendes, J. F. (2013). Evolution of networks: From biological nets to the Internet and WWW. OUP Oxford.

[3] Girvan, M. & Newman, M. E. (2002). Community structure in social and biological networks, Proceedings of the national academy of sciences, vol. 99, no. 12, pp. 7821-7826.

[4] Fortunato, S. (2010). Community detection in graphs, Physics reports, vol. 486, no. 3-5, pp. 75-174.

[5] Berahmand, K., Bouyer, A. & Vasighi, M. (2018). Community Detection in Complex Networks by Detecting and Expanding Core Nodes Through Extended Local Similarity of Nodes, IEEE Transactions on Computational Social Systems, vol. 5, no. 4, pp. 1021-1033.

[6] Fanrong, M., Mu, Z., Yong, Z. & Ranran, Z. (2014). Local community detection in complex networks based on maximum cliques extension, Mathematical Problems in Engineering, vol. 2014.

[7] Raghavan, U. N., Albert, R. & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks, Physical review E, vol. 76, no. 3, pp. 036106.

[8] Tasgin, M. & Bingol, H. O. (2019). Community detection using boundary nodes in complex networks, Physica A: Statistical Mechanics and its Applications, vol. 513, pp. 315-324.

[9] Nourani, E., zarezadeh, M. & Bouyer, A. (2019). Identifying Communities Using Label propagation of Boundary Nodes and local similarity measures, in 4th Conference on Electerical and Computer Engineering Technology, Tehran, Iran, 2019: ISC, pp. 1-8.

[10] Lancichinetti, A. & Fortunato, S. (2012). Consensus clustering in complex networks, Scientific reports, vol. 2, pp. 336.

[11] Ashrafi Payaman, N. & Kangavari, M. R. (2018). Graph Hybrid Summarization, Journal of AI and Data Mining, vol. 6, no. 2, pp. 335-340.

[12] Blondel, M. R. , Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. (2008). Fast unfolding of communities in large networks, Journal of statistical mechanics: theory and experiment, vol. 2008, no. 10, pp. P10008.

[13] Newman, M. E. (2004). Fast algorithm for detecting community structure in networks, Physical review E, vol. 69, no. 6, pp. 066133.

[14] Jiang, J. Q., Dress, A. W. & Yang, G. (2009). A spectral clustering-based framework for detecting community structures in complex networks, Applied Mathematics Letters, vol. 22, no. 9, pp. 1479-1482.

[15] Lambiotte, R., Delvenne, J.-C. & Barahona, M. (2008). Laplacian dynamics and multiscale modular structure in networks, arXiv preprint arXiv:0812.1770.

[16] Lai, D., Nardini, C. & Lu, H. (2011). Partitioning networks into communities by message passing, Physical review E, vol. 83, no. 1, pp. 016115.

[17] Berahmand, K. & Bouyer, A. (2019). A Link-Based Similarity for Improving Community Detection Based on Label Propagation Algorithm, Journal of System Science and Complexity, vol. 32, no. 3, pp. 737–758.

[18] Berahmand, K. & Bouyer, A. (2018). LP-LPA: A link influence-based label propagation algorithm for discovering community structures in networks, International Journal of Modern Physics B, vol. 32, no. 06, pp. 1850062.

[19] Pan, Y., Li, D.-H., Liu, J.-G. & Liang, J.-Z. (2010). Detecting community structure in complex networks via node similarity, Physica A: Statistical Mechanics and its Applications, vol. 389, no. 14, pp. 2849-2857.

[20] Liu, C., Liu, J. & Jiang, Z. (2014). A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks, IEEE transactions on cybernetics, vol. 44, no. 12, pp. 2274-2287.

[21] Sun, P. G., Gao, L. & Han, S. S. (2011). Identification of overlapping and non-overlapping community structure by fuzzy clustering in complex networks, Information Sciences, vol. 181, no. 6, pp. 1060-1071.

[22] Karimi Zandian, Z. & Keyvanpour, M.- R. (2019). MEFUASN: A Helpful Method to Extract Features using Analyzing Social Network for Fraud Detection, Journal of AI and Data Mining, vol. 7, no. 2, pp. 213-224.

[23] Lou, H., Li, S. & Zhao, Y. (2013). Detecting community structure using label propagation with weighted coherent neighborhood propinquity, Physica A: Statistical Mechanics and its Applications, vol. 392, no. 14, pp. 3095-3105.

[24] Gregory, S. (2010). Finding overlapping communities in networks by label propagation, New Journal of Physics, vol. 12, no. 10, pp. 103018.

[25] Lin, Z., Zheng, X., Xin, N. & Chen, D. (2014). CK-LPA: Efficient community detection algorithm based on label propagation with community kernel, Physica A: Statistical Mechanics and its Applications, vol. 416, pp. 386-399.

[26] Zachary and W. W. (1977). An information flow model for conflict and fission in small groups, Journal of anthropological research, vol. 33, no. 4, pp. 452-473.

[27] Lusseau, D., Schneider, K., Boisseau, O., Haase, P. J., Slooten, E. & Dawson, S. M. (2003). The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations, Behavioral Ecology and Sociobiology, vol. 54, no. 4, pp. 396-405.

[28] Newman, M. E. & Girvan, M. (2004). Finding and evaluating community structure in networks, Physical review E, vol. 69 , no. 2, pp. 026113.

[29] Leskovec, J. & Krevl, A. SNAP Datasets: Stanford large network dataset collection [Online] Available: http://snap.stanford.edu/data

[30] You, X., Ma, Y. & Liu, Z. (2019). A three-stage algorithm on community detection in social networks, Knowledge-Based Systems.

[31] Zarandi, F. D. & Rafsanjani, M. K. (2018). Community detection in complex networks using structural similarity, Physica A: Statistical Mechanics and its Applications, vol. 503, pp. 882-891.